Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Internet of Things 12 (2020) 100297

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Locally reconfigurable Self Organizing Feature Map for high impact malicious tasks submission in Mobile Crowdsensing☆

Xuankai Chen, Murat Simsek, Burak Kantarci*

School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Ave,Ottawa, ON K1N 6N5, Canada

ARTICLE INFO

Article history:
Received 31 March 2020
Revised 27 August 2020
Accepted 12 September 2020
Available online 29 September 2020

Keywords:
Mobile Crowdsensing
Artificial neural networks
Self-Organizing Feature Map
Fake task submission
Clogging attacks

ABSTRACT

Location-based clogging attacks in a Mobile Crowdsensing (MCS) system occur following upon the submission of fake tasks, and aim to consume the batteries and hardware resources of smart mobile devices such as sensors, memory and processors. Intelligent modeling of fake task submissions is required to enable the development of effective defense mechanisms against location-based clogging attacks with fake task submissions. An intelligent strategy for fake task submission would aim to maximize the impact on the participants of an MCS system. With this in mind, this paper introduces new algorithms exploiting the Self-Organizing Feature Map (SOFM) to identify attack locations where fake sensing tasks submitted to an MCS platform are centered around. The proposed SOFM-based model addresses issues in the previously proposed SOFM-based attack models by proposing two ways of refinement. When compared to the former models, which also use SOFM architectures, simulation results show that up to 139.9% of impact improvement can be modeled under the reconfigurable SOFM architectures.

1. Introduction

Smart mobile devices such as smartphones, wearables and tablets are equipped with various types of sensors making the smart mobile devices ideal sources of collecting a variety of data [1]. Mobile Crowdsensing (MCS) is a community sensing mechanism introduced in [2] and further discussed in [3,4]. In a participatory MCS model, users proactively contribute data to an MCS system and gain rewards by completing certain data collection tasks. This, in turn, empowers the construction of a smart city [5–7]. For instance, tasks deployed around a busy intersection in a city require users to take photos of the intersection so that the city administration can make suggestions regarding the traffic condition at different times throughout the day [8].

Research on the security and privacy of MCS has been conducting actively since MCS systems can be targets of attacks in various aspects [9,10]. Submission of altered/fake sensed data has been well investigated under the data poisoning attacks in MCS systems [11] as adversaries can inject fake/altered data in response to sensing tasks in order to gain rewards. However, besides the data poisoning, malicious users in an MCS system perform Denial-of-Service (DoS) attacks by continuously injecting invalid data into the system [12]. Numerous incentive mechanisms have been proposed to mitigate the

☆ This work was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) under the DISCOVERY Program.
* Corresponding author.
E-mail address: burak.kantarci@uottawa.ca (B. Kantarci).

https://doi.org/10.1016/j.iot.2020.100297
2542-6605/

impact of attacks by malicious users [13,14]. Particular focuses of user incentives have been on truthfulness of participation [15], security and privacy of crowdsensed data [16], game theoretic approaches to ensure reliability and dependability of crowdsensing services [17,18].

As stated in [19], batteries are critical components of MCS systems whereas sensing, processing and storage resources of mobile devices are of the paramount importance as well. Given these, since users are allowed to create and submit sensing tasks to an MCS platform, MCS platforms and participants are vulnerable to Denial of Service (DoS)-like attacks initiated by malicious users who submit fake tasks which interfere with the normal execution of other tasks in the same MCS system [20]. Due to the unavailability of data regarding fake task submissions, it is imperative to design synthetic traces to model the behaviour of malicious users who submit fake tasks to clog the system resources in an MCS network. The synthetic model should be designed by considering the broadest impact of a submitted fake task on an MCS system. Therefore, adversaries are modeled for submitting their fake tasks to have the epicenter of a task cover as many participants as possible. To the best of our knowledge, the study in [20] considered the threats concerning DoS-like clogging attacks on MCS systems for the first time, and introduced an empirical approach to model the adversarial behavior in the presence of such attacks. The authors defined the following key features that characterize the behaviour of a sensing task: location, time, resource requirements (e.g. battery drain). Later in [21,32], an Artificial Intelligence-based approach was proposed by training a Self-Organizing Feature Map (SOFM) to determine the sensing task deployment locations alongside the other key features to maximize the impact of a fake sensing task. These studies were followed by defence mechanisms to detect and eliminate the sensing tasks by mainly using ensemble machine learning methods [22,23]. Recently, SOFM was used as an AI engine to determine the best position for the early flattening of COVID-19-like pandemics in [33]. The position was utilized as a center of the region that the mobile assessment center is assigned to test all individuals located in that region. Moreover, all stops of a mobile assessment center considering the worst-case scenario were optimally obtained by SOFM for each region in the selected area.

This paper builds on the previously introduced SOFM-based adversarial model and proposes a locally reconfigurable SOFM architecture for higher impact on the participant population. Each neuron of the SOFM denotes an attack center for the fake sensing task submissions to the MCS platform. The proposed approach pursues a refinement procedure for the SOFM after training with participant coordinates. The training period is followed by a posterior search over an extended region to determine the locations of the fake tasks. Simulation results show that the proposed locally reconfigurable SOFM can achieve up to 139% increase in terms of the mean estimate of the number of impacted user events (defined in Section 4.3.3).

Compared to Zhang et al. [21], this paper aims to solve a similar problem with several improved methods and combinations of them so to overcome the bottleneck of the method proposed in [21], (which is described in Section 2.3) and significantly increase the impact of adversaries in some cases.

While this study models an adversary, the study in [22] models a defender. An important objective of fake task generation is to disguise the fake tasks, so that they are not easily notified and avoided.

The rest of the paper is organized as follows. In Section 2, background and motivation of this work is presented. Section 3 presents in detail the two proposed approaches to improve the performance of attack center generation. In Section 4, the results of different approaches with several metrics are compared. Section 5 concludes on the effect of the proposed approaches and suggest on the direction for further improvement.

2. Background and motivation

This section starts with presenting the background information on Self-Organizing Feature Map and related studies in MCS. Following that, the motivation behind this work is discussed, where insights are provided to motivate the proposed approach.

2.1. Self-Organizing Feature Map (SOFM)

Self-Organizing Feature Map (SOFM) is the core of the proposed model in this paper, and it is an Artificial Neural Network (ANN) introduced in [24] and further revisited in various studies [25–27].

From a practical perspective, SOFM has been studied in [28], and recently in [33,34]. It is worth mentioning that problems studied in [33,34] are particularly related to the problems discussed in this paper. The authors in [33] leverage SOFM to optimally deploy and route the mobile agents in the case of a pandemic by making an analogy to the problem of optimal placement of adversarial tasks in an MCS setting.

Basically, SOFM is used for the recognition of the spatial distribution of a set of data with a network of neurons. Below are the steps of the SOFM technique and the settings used in this paper. It is worth to note that for the sake of simplicity, only the basic version of SOFM is presented.

(1) Initialization of the network: the initial network can be formed in the shape of a hexagon or rectangle.
In this paper, two features of user events are used in order to train SOFMs. These features are the longitude and latitude. The input vector of the SOFM training is

$$\mathbf{x} = \left[\mathbf{x}_i\right]_{n\times 2}, i = 1, \ldots, n,$$

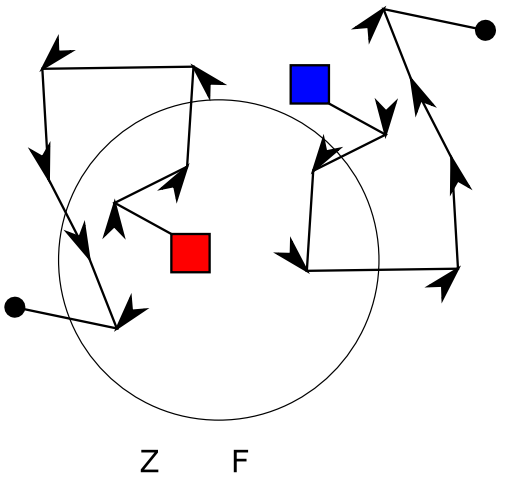


Fig. 1. Two movement models that were previously proposed. Red and blue squares represent the start locations of a malicious task and a legitimate task, respectively, whereas the black paths with arrows represent the random walks movement pattern of tasks during the MCS campaign. The black dots are the last positions of a task. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where $\mathbf{x}_i = \left(x_{i,\text{ longitude}}, x_{i,\text{ latitude}}\right)$ is the geographic coordinate of the ith user event and n is the number of user events. A $p \times q$ SOFM is initialized uniformly in the minimum bounding rectangle of the city map. The neuron vector (or weight vector) is denoted by

$$\mathbf{w} = \left[\mathbf{w}_{ij}\right]_{p\times q\times 2}, i = 1, \ldots, p, j = 1, \ldots, q.$$

(2) Selection of the Best-Matching Unit (BMU, also called Best-Matching Cell in [25]): a BMU is the "winner" neuron for an input in the current SOFM, which is the closest SOFM neuron based on the Euclidean distances.

(3) Updating of the weight vector ($\mathbf{w}$): at each iteration of the training, the algorithm selects an input ($\mathbf{x}_k$), and finds the corresponding BMU ($\mathbf{w}_{(k)}$). It then defines a radius and updates the neighboring neurons of $\mathbf{w}_{(k)}$ in the current SOFM within the range of the radius. The updating process of a neuron can be formulated as

$$\forall \mathbf{w}_i \in \mathcal{N}\left(\mathbf{w}_{(k)}\right), \mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} + \alpha_{i,k}^{(t)}\left(\mathbf{x}_k^{(t)} - \mathbf{w}_i^{(t)}\right),$$

where $\mathcal{N}\left(\mathbf{w}_{(k)}\right)$ is the set of neighboring neurons of $\mathbf{w}_{(k)}$ and $\alpha_{i,k}^{(t)}$ is a scalar value ranging between 0 and 1. The study in [27] suggests a feasible choice of $\alpha_{i,k}^{(t)}$, which is

$$\alpha_{i,k}^{(t)} = c(t)\exp\left\{-\frac{\text{dist}^2\left(\mathbf{w}_i, \mathbf{w}_{(k)}\right)}{2\sigma^2(t)}\right\},$$

where $\text{dist}\left(\mathbf{w}_i, \mathbf{w}_{(k)}\right)$ is the Euclidean distance between $\mathbf{w}_i$ and $\mathbf{w}_{(k)}$, $c(t)$ and $\sigma(t)$ are two monotonically decreasing functions of t, and the initial value of $\sigma(t)$ is sufficiently large.

2.2. Determination of attack centers with SOFM

According to Zhang et al. [21], attack centers are first determined based on the density of user events in a city, and malicious tasks are then generated within a defined radius of attack centers. Similar to the legitimate tasks, malicious tasks can move over the time. Two movement models, namely Zone-Free Movement (ZFM) and Zone-Limited Movement (ZLM), are introduced in [20] and [22], respectively [21].

Fig. 1 illustrates the ZFM based legitimate and illegitimate tasks movements. In a deployment with ZFM, malicious tasks are initially generated within a defined radius and will be able to move outside the radius afterwards, while legitimate tasks are initially generated outside of an attack region, then they can move inside of the attack region according to a random walk movement pattern. All movements with ZLM are constrained within the radius. As ZFM does not restrict the movement of tasks in either the inbound or outbound direction, it is considered to be more realistic than ZLM. Therefore ZFM is selected for the task movements in this paper.

To find the best attack centers, the study in [21] follows an Artificial Neural Networks-assisted approach, where SOFMs are trained. The best neurons will be selected as the attack centers.

Algorithm 1 describes the procedure followed in [21].

Algorithm 1 Find attack centers with SOFM (Original method in [21]).

```
procedure FindAttackCenters(AllUserEvents)
    BestNeurons ← ∅
    for all day do
        UserEvents ← GetUserEvents(AllUserEvents, day)
        SOFM ← TrainSOFM(UserEvents)
        ▷ Neurons ranked by the number of user events covered
        BestNeurons ← BestNeurons ∪ {GetBestNeuron(SOFM)}
    end for
    return BestNeurons
end procedure
```

2.3. Motivation

In [21], all SOFMs are trained with global data in a city. While a SOFM is being trained, the position of a single neuron is likely to be affected by the positions of other neurons. This can be told from Step (3) in Section 2.1, where neighbouring neurons will be updated at each iteration. It is observed that in some cases, a neuron in a trained SOFM does not reflect a locally dense area. In other words, a denser area could have been covered if the neuron were shifted.

The network size of SOFM is based on the number of neurons which is determined at the beginning of the training process. Increasing the network size will lead to a decrease in the number of covered individuals per neuron. Applying constraints to the neuron coverage is a complicated process, which is the reason why a more detailed solution can be obtained by the regional-based selection of the number of neurons as reported in [33]. During the proposed refinement process, any possible improvement can be applied to each neuron through local SOFM update as depicted in Fig. 2.

3. Proposed solutions

In this section, two approaches are proposed to improve the performance (defined in Section 4.1) of attack location generation. These two variants of SOFM are collectively referred to as "locally reconfigurable SOFM."

The key assumption in this paper is as follows: a user can execute no more than one task at any moment. It is further assumed that a user event will be contracted by an illegitimate task if possible and by the task that leads to the highest energy consumption.

3.1. Refinement of SOFM neurons with locality

As mentioned in Section 2.3, the SOFM-base approach proposed in [21] might neglect the locally dense areas when SOFMs are trained. Hence, the first improvement this paper puts forward is to refine neurons of SOFMs with the consideration of locality.

An example is shown in Fig. 4.

Fig. 4 (a) shows a neuron in an initially trained SOFM and user events covered within a 200 m radius. Apparently, the coverage of the neuron could be improved. The neuron is refined by defining an outer radius (300 m in this case) and retraining a 1×1 SOFM with the user events happening within the outer circle (other centroid-based clustering algorithms to find a denser area can also be considered), shown in Fig. 4(b). The result of the refinement is shown in Fig. 4(c), where the blue point and the blue circle are the refined neuron and the new coverage, respectively. The procedure is defined in detail in Algorithm 2.

Different from Algorithm 1, Algorithm 2 attempts to achieve a better performance and enlarge the coverage of user events by slightly moving each initially generated attack center within a certain bound. After obtaining the performance statistics (i.e., the number of covered user events) of the adjusted attack locations, the algorithm ranks the statistics to determine its choice of attack locations, similar to that in Algorithm 1.

3.2. Distance constraint for neuron selection

In the original approach [21] and the refined approach in Section 3.1, the best neurons are selected one by one, without using the knowledge of any previously selected neurons. It is possible that two selected attack centers are too close as shown in the example scenario in Fig. 3.

Under the assumption explained before Section 3.1, a user can only execute at most one task at a time. When illegitimate tasks are deployed in the overlapping area of the coverage of two close attack locations, the potential impact of individual

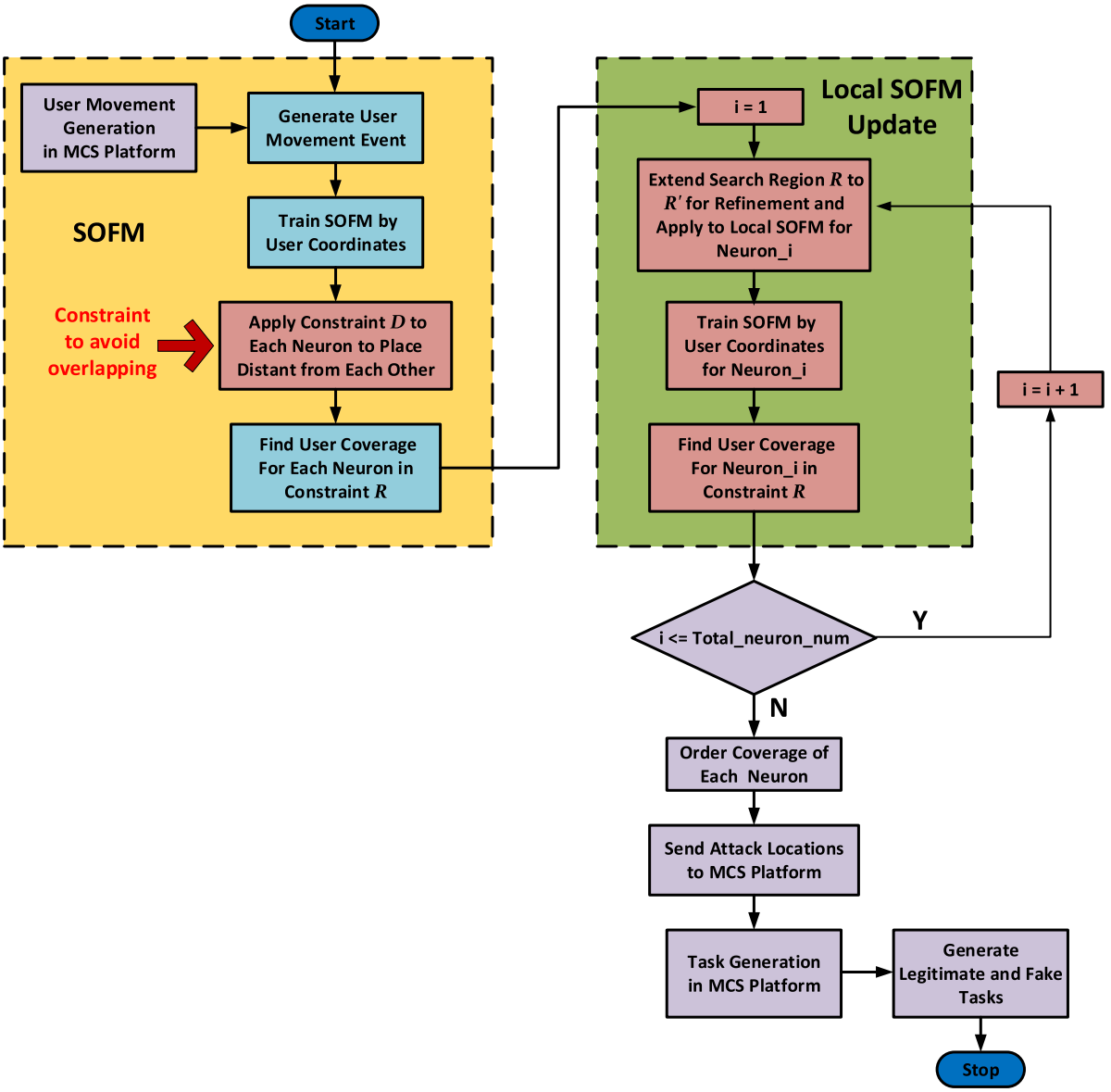


Fig. 2. Reconfigurable SOFM implementation to MCS platform where *i* and *Total_neuron_num* indicate the neuron index and the number of neurons in the SOFM configuration, respectively.

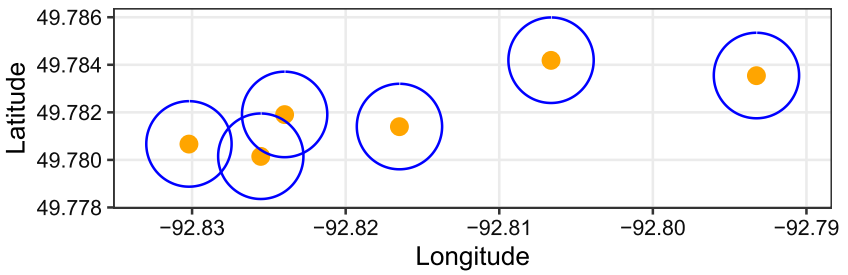
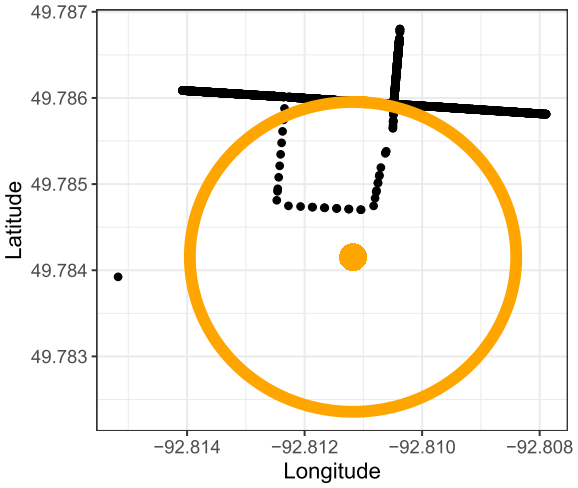
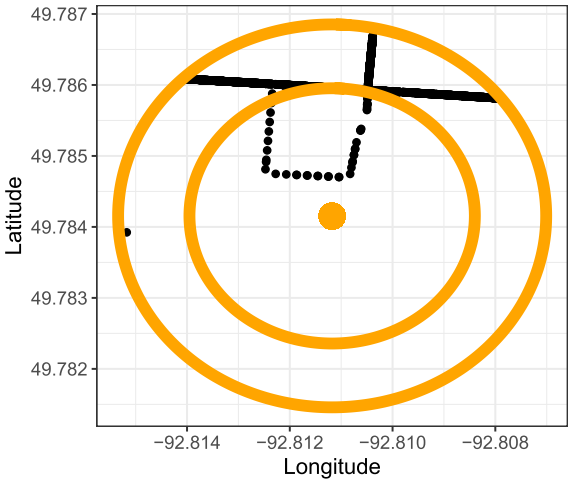


Fig. 3. Three close neurons are selected as attack centers. The coverage of illegitimate tasks may overlap when deployed within a 200 m radius of these centers.



(a) Coverage of a neuron before refinement



(b) User events included for retraining SOFM

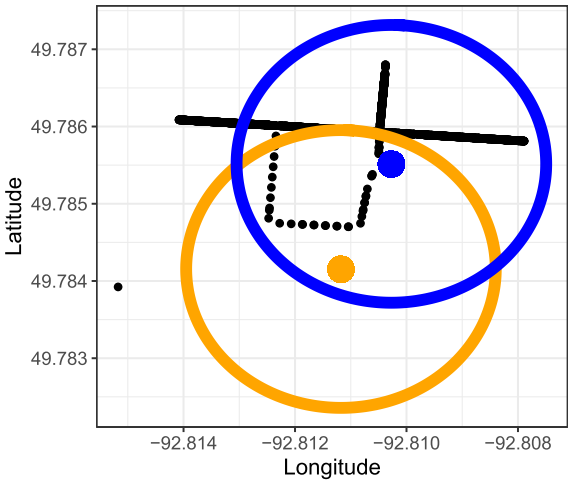


Fig. 4. Example refinement result for a SOFM neuron.

Algorithm 2 Find attack centers with refined SOFM.

```
R′: Extended search region for refinement
procedure FindRefinedAttackCenters(AllUserEvents)
    OldBestNeurons ← FindAttackCenters(AllUserEvents)
    NewBestNeurons ← ∅
    for all day do
        UserEvents ← GetUserEvents(AllUserEvents, day)
        SOFM ← TrainSOFM(UserEvents)
        NewNeurons ← ∅
        for all OldNeuron ∈ SOFM do
            CoveredUserEvents ← GetCoveredUserEventsInRange(OldNeuron, R′)
            NewNeurons ← NewNeurons ∪ TrainSOFM(CoveredUserEvents)        ▷ Train 1 × 1 SOFM
        end for
        ▷ Neurons ranked by the number of user events covered
        NewBestNeurons ← NewBestNeurons ∪ {GetBestNeuron(NewNeurons)}
    end for
    return NewBestNeurons
end procedure
```

tasks could be reduced. To avoid such a situation, we apply distance constraints when picking the best attack centers, which can be formulated as shown in Eq. (1).

$$\text{geodesic}\left(c_i, c_j\right) \geqslant 400\text{m}, \forall c_i, c_j \in \mathcal{C}, c_i \neq c_j, \tag{1}$$

In the equation, $\mathcal{C}$ is the set of selected attack centers, and $\text{geodesic}\left(c_i, c_j\right)$ is the geodesic distance[1] [29] between c_i and c_j.

This work benefits from a feature of SOFM which is that many candidate attack centers are available, so it is unlikely to run out of neurons with the constraint in Eq. (1).

Algorithm 3 presents the selection process presuming that we have access to neurons in all SOFMs trained by Algorithms 1 or 2.

Algorithm 3 Select attack locations with distance constraints.

```
D: Isolation constraint between neurons
procedure SelectAttackLocationsWithDistanceConstraints(Candidates)
    BestNeurons ← ∅
    Candidates ← Rank(Candidates)        ▷ Neurons ranked by the number of user events covered
    for all c_i ∈ Candidates do
        for all c_j ∈ BestNeurons do
            if geodesic(c_i, c_j) ⩾ D then
                BestNeurons ← BestNeurons ∪ {c_i}
                break        ▷ Found one satisfying the constraint. Continue with the next candidate
            end if
        end for
    end for
    return BestNeurons
end procedure
```

4. Performance evaluation

As previous research in [21] showed that effective to utilization of SOFMs could determine attack centers, this paper uses the SOFM-based attack center determination as a benchmark solution. With this in mind, CrowdSenSim simulator [30,31] is utilized to generate tasks and participant mobility patterns. The pipeline of the simulation environment built on CrowdSenSim is illustrated in Fig. 2.

[1] GeoPy Documentation, https://geopy.readthedocs.io/en/stable/#module-geopy.distance

Table 1
Simulation settings for user events, attack center and task generation. Uniform(x,y) denotes a uniform distribution of a variable in the interval (*x*,*y*).

Attack region radius (R_a)	200 m
Number of users	10, 000
Duration of user events	Uniform(60min, 180min)
Smartphone battery level (user)	80% from Uniform(20%, 100%); 20% from Uniform(1%, 20%)
Sample rate of user events	1 event/min
Daily simulation time interval	00:00 – 23:59
Number of attack centers	6
Number of tasks	{500, 1000, 2000}

Table 2
Simulation settings for task event generation.

	Malicious tasks	Legitimate tasks
Day	Uniform($\{1, 2, \ldots, 6\}$)	
Hour	80% in Uniform(7:00, 11:00); 20% in Uniform(12:00, 17:00)	Uniform(00:00, 23:59)
Duration	70% from Uniform({40min, 50min, 60min}); 30% from Uniform({10min, 20min, 30min})	Uniform({10min, 20min, ..., 60min})
Smartphone battery usage	50% from Uniform(7%, 10%); 50% from Uniform(1%, 6%)	Uniform(1%, 10%)
Coverage	100m	
Movement radius	80m	

4.1. Performance metrics

The following definitions help to understand the performance evaluation study in this section.

- Impacted user events: the number of distinct user events covered by malicious tasks. A user event (U) is said to be covered by a task event (T) if all of the following conditions are met: (1) the timestamp of U is within the time interval of T; (2) the location of U is within the coverage of T, which means geodesic$(U, T) \leqslant$ 100m in our experiments; (3) the battery of the user device at the moment of U is sufficient for the corresponding task of T; and (4) U has not been covered by other task events.
- Impacted users: the number of distinct users covered by malicious tasks, which is obtained from the unique user IDs of the impacted user events.
- Completed malicious tasks: the number of distinct malicious tasks completed by users. A task is completed if there is any task event that covers at least one user event.
- Total impacted energy consumption: the total value of energy consumed by all malicious task events that cover user events.
- Ratio of impacted user events: the ratio of the number of total impacted events to the number of total events.
- Ratio of impacted energy consumption: the ratio of total impacted energy consumption to the total amount of energy consumption.

4.2. Simulation setup

This paper adopts the performance evaluation settings used in [20,21], detailed in Table 1 for the generation of user events, attack centers and tasks.

Key parameters in Table 1 are explained as follows.

- Attack region radius (R_a): the maximum distance between the generated malicious tasks and the corresponding attack center.
- Duration of user events: the amount of time a user event lasts for before another user event starts. It can be interpreted as a rate of sampling.
- Smartphone battery level (user): the battery level of a user device captured at each user event.

In addition to these, Table 2 presents the performance evaluation settings for the generation of task events.

To maximize the impact, illegitimate tasks are mainly deployed during on-peak hours[2], the time when users are active the most within a day. Specifically, 80% of the illegitimate tasks are active during on-peak hours, while the remaining 20% aim for the mid-peak hours, and no illegitimate tasks are submitted during off-peak hours since the adversaries are unlikely to gain much due to the overall reduced user activity.

[2] We follow the schedules for on-peak and mid-peak hours given by the Ontario Energy Board [35], since these schedules would reflect the patterns of electricity consumption of users and the activeness of users in a day.

With the limited resources, it is also impossible to keep all tasks long-lasting. Simulations in this paper assume that only 70% of the illegitimate tasks run for over 40 min, and 30% last for less than 30 min. Furthermore, the duration is assumed to be discrete for the sake of simplicity in simulation and analysis. Last but not least, half of the illegitimate tasks are assumed to use 7% to 10% of the smartphone battery during an attack on a device, while the other half are assumed to consume only 1% to 6%. These assumptions are adopted from previous works in [22,34].

Tasks, users and smartphone data are generated in the following format: *task* = {'ID', 'latitude', 'longitude', 'day', 'hour', 'minute', 'duration', 'remaining time', 'battery consumption (%)', 'legitimacy', 'on-peak hour', 'grid number'}. *user* = {'ID', 'latitude', 'longitude', 'day', 'hour', 'minute', 'moving duration'}. *smart phone* ={'ID', 'UserID', 'battery level (%)', 'sensing radius'}. Among these features, the location of a user or a task is determined by latitude, longitude, and time of its presence consisting of day, hour and minute. Particularly, the duration for a task is the time to accomplish the whole task and the remaining time exhibits the time needed to finish the subsequent sub-tasks. Battery consumption denotes the battery level of smart phone percentage level in the format of the percentage requested by the task. The moving duration for a user indicates the temporal length of the motion of the user. As a Wi-Fi router operating at 2.4 GHz band can reach up to 300 feet outdoors, the sensing radius is set at 100 m. Illegitimate tasks from clogging attackers constitute 10% of total tasks. 'Legitimacy' is the class label to differentiate fake tasks from legitimate tasks. Moreover, 'on-peak hour' which is a Boolean feature is utilized to indicate whether the task is initiated during 7am to 11am which is defined as busy communication time. The last feature "grid number" is added to quantify the coordinates. Rectangular partitions, which are based on the minimum latitude, minimum longitude, maximum latitude and maximum longitude of the city map, divide a city into small grids with approximately the length of 600 m. Labels start with one and increase towards the last partition according to the matrix structure, ascending from east to west and from north to south.

4.3. Results

MCS platform is used for data aggregation which consists of user movements for six days and task generation for twenty days in order to implement SOFM and locally reconfigurable SOFM proposed in this work. User movement data for six days are evaluated by SOFM to determine the attack centers with different strategies. Legitimate and illegitimate tasks are generated through MCS platform and fixed for evaluation of efficiency. The twenty days of user movement data, which are generated independently from the six-day user movement data, are utilized to demonstrate more challenging extrapolation performance for the test performances of SOFM and the proposed locally reconfigurable SOFM. Performance evaluation is implemented in two cities, specifically Dryden as a small city and Brant as a larger city, for us to consider a more generalized case study.

Since two approaches are proposed for improvement, there are four combinations of the approaches, which are

- not refined and not constrained (baseline),
- refined and not constrained,
- not refined and constrained,
- refined and constrained.

Refinement is briefly explained as local update around each neuron after SOFM training phase as depicted in Fig. 4 and also followed in Fig. 2. The number of potential participants in a 200 m radius of the attack center are calculated for all SOFM neurons, then an attack center satisfying the maximum number of participants is determined as an illegitimate task submission region for more intelligent and realistic attack scenarios. If two neurons are closer than 400 m as depicted in Fig. 3, they are overlapped each other and the performance of SOFM is mitigated in this case. This issue can be solved by applying a 400 m constraint to assure that each neuron has its own coverage and each participant can be covered by only a single SOFM neuron. After implementing these two mechanisms in SOFM, the locally reconfigurable SOFM provides better coverage performance to determine attack center. All improvements after applying refinement and constraint are summarised in Table 3 and Fig. 5 for Brant. The overall selected neurons by a reconfigurable SOFM for attack center have a better coverage in terms of the number of potential participants than those by SOFM.

Afterwards, we aggregate data to obtain the performance metrics that were described in Section 4.1, and perform a *t*-test to get a mean estimate and a 95% confidence interval for each performance metric. The results for Dryden and Brant are shown in Figs 7 and 6, respectively.

In each group (separated by the number of tasks) of a sub-figure in Figs 7 and 6, the bars from left to right represent the results in the above order. Below we provide individual discussions for these results concerning the refined and constrained approaches, as well as the combination of these two approaches.

4.3.1. Impact of refined SOFM

As seen in Fig. 7, the SOFM-based modelling of fake tasks after refinement (i.e. "Refined, Unconstrained" in the figure) leads to little difference for Dryden in the number of completed malicious tasks, the value of the impacted energy consumption, and the fraction of impacted energy. On the other hand, in terms of the other performance metrics, remarkable improvements are achieved. For instance, the number of impacted users, impacted user events, and the fraction of impacted user events increase by around a factor of one when 2000 tasks are deployed.

Table 3
Changes of neuron coverage after refinement in Brant. Coverage of all neurons has been improved. Neurons in bold font are selected since they lead to the highest coverage of user events after refinement.

Neuron	User events covered by original neuron (circle in Fig. 4(a), inner circle in Fig. 4(b) or orange circle in Fig. 4(c))	User events within 300 m of original neuron (outer circle in Fig. 4(b))	User events covered by refined neuron (black points within blue circle in Fig. 4(c))	Increase	Percentage of increase
1	0	6	8	8	*NaN*
2	86	212	181	95	110%
3	659	1118	819	160	24%
4	106	190	122	16	15%
5	0	459	492	492	*NaN*
6	3220	5235	3292	72	2%
7	1618	3510	2199	581	36%
8	333	575	384	51	15%
9	381	635	418	37	10%
10	199	410	307	108	54%
11	953	2173	1158	205	22%
12	311	645	508	197	63%

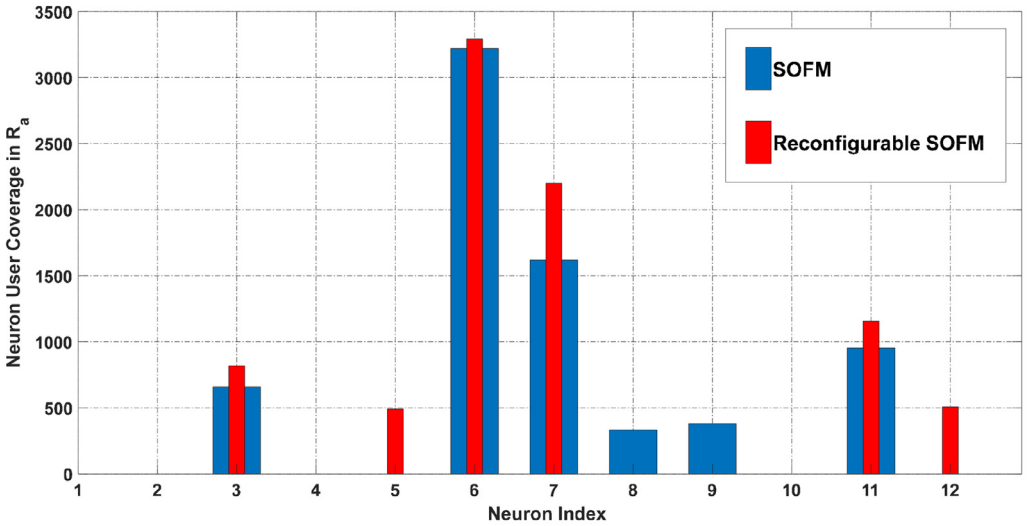


Fig. 5. The number of covered users under SOFM and Reconfigurable SOFM.

As seen in Fig. 7, refined SOFM-based modelling (i.e. "Refined, Unconstrained" in the figure) of fake tasks in a larger city (i.e. Brant) leads to significant improvements in most performance metrics and a slight degradation in the number of participants and the value of energy consumption for 500 tasks. Furthermore, the improvements become more significant as the number of tasks increases. For instance, we observe a 24.7%, 25.1%, and 34.1% improvements in the mean estimate for the impacted energy consumption for 500, 1000, and 2000 tasks, respectively.

To evaluate the impact of refinement, we also examine the number of user events covered by individual neurons at each stage during the training phase. To this end, changes of the numbers of user events covered by individual neurons in Brant are shown in Table 3.

4.3.2. Impact of constrained SOFM

As seen in Figs 7 and 6, the constrained SOFM ("unrefined, constrained" in the figure) can also contribute to the improvement, but it is not as substantial as the refined approach, and some degradation can be noticed in some cases. Similar to the refined approach, the most remarkable improvement can be seen when measured with the number of impacted users and impacted user events in Dryden. After the application of the constrained approach, the number of impacted users goes up by 33.5%, 61.1%, and 54.8%, for 500, 1000, and 2000 tasks, respectively. The corresponding increases for the number of impacted user events are 45.7%, 69.3%, and 69.2%.

4.3.3. Impact of refined and constrained reconfigurable SOFM

When two approaches are applied consecutively, i.e., applying the refined approach during the training process and distance constraints while selecting the attack centers, the impact is more significant than the experiments when a single approach is applied. This scheme is denoted by "refined, constrained" in Figs 7 and 6.

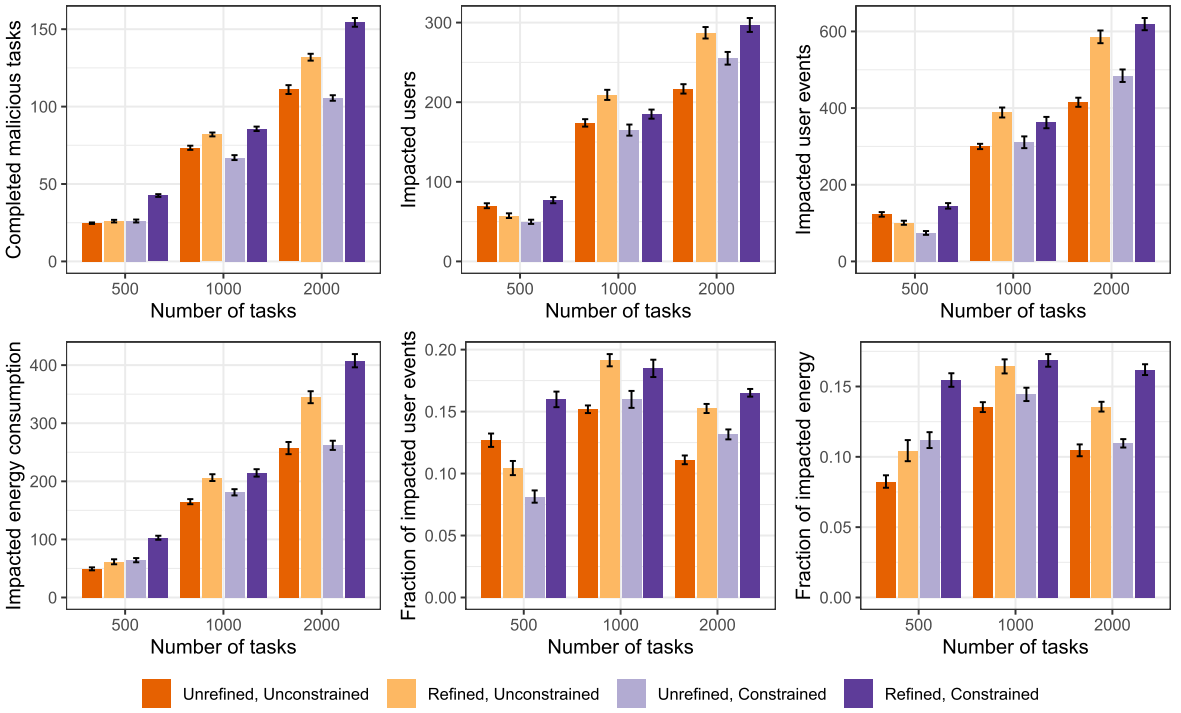


Fig. 6. Performance of reconfigurable SOFM in Brant: completed malicious (illegitimate) tasks, impacted users, impacted user events, impacted energy consumption, fraction of impacted user events, and fraction of impacted energy.

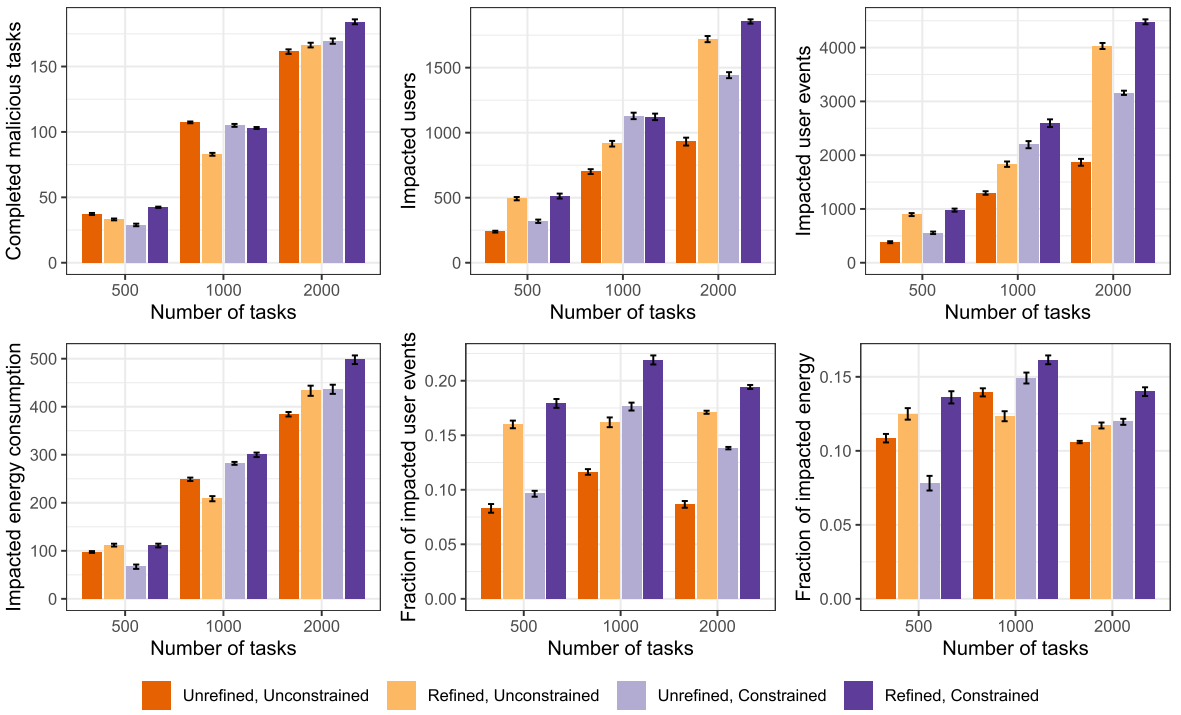
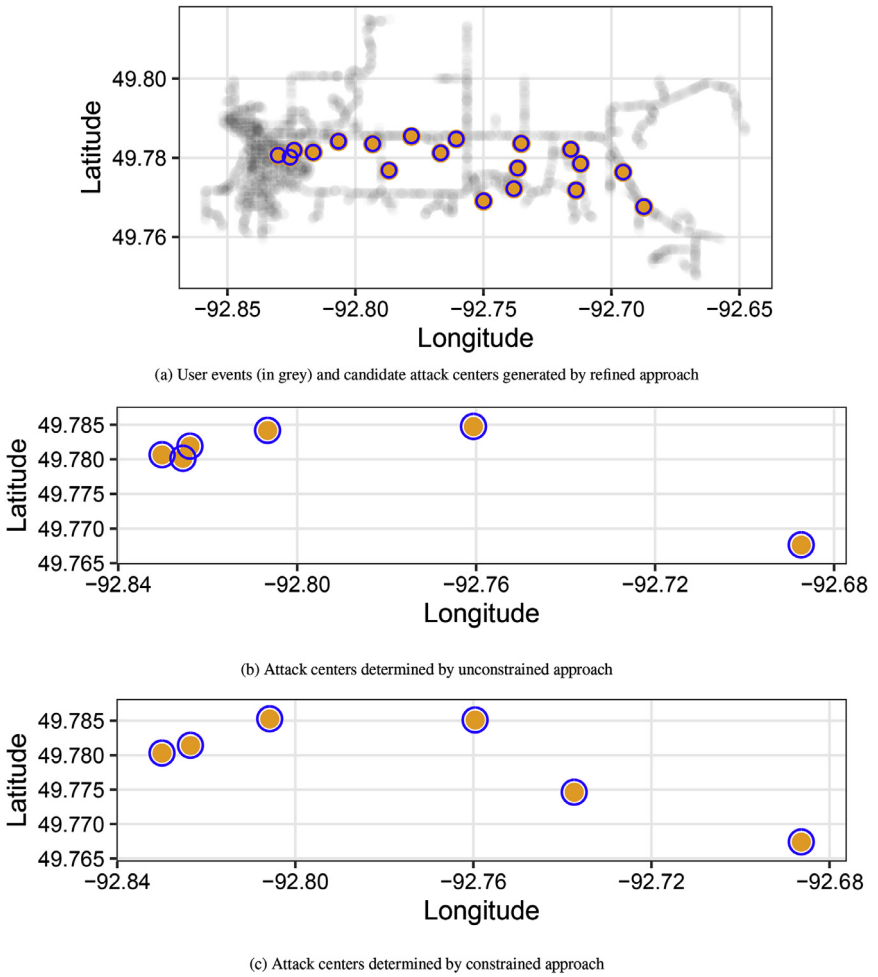


Fig. 7. Performance of reconfigurable SOFM in Dryden: completed malicious (illegitimate) tasks, impacted users, impacted user events, impacted energy consumption, fraction of impacted user events, and fraction of impacted energy.



(a) User events (in grey) and candidate attack centers generated by refined approach

(b) Attack centers determined by unconstrained approach

(c) Attack centers determined by constrained approach

Fig. 8. Determination of attack centers in Dryden with both unconstrained and constrained approaches. Orange points represent attack centers and blue circles are the corresponding attack regions (R_a). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

For example, compared to the baseline approach, the refined approach, and the constrained approach, the number of impacted users with a total of 2000 tasks in Dryden is increased by 99.2%, 7.9%, and 28.6%, respectively. The corresponding boost for the number of impacted user events are 139.9%, 11.2%, and 41.8%, respectively. These results consolidate the implication that the refined approach is more effective than the constrained approach, from Section 4.3.2.

4.3.4. t-SNE analysis on generated tasks

Since this study models an adversary, it needs to ensure that the generated illegitimate tasks are effectively disguised, which means that they should be analogous to legitimate tasks. To demonstrate the similarity, t-distributed stochastic neighbor embedding (t-SNE) analysis is performed, which is a dimension reduction technique proposed in [36] to visualize the clusters of data. Here, the tasks generated for Brant are considered, and the closeness of tasks is measured based on a subset of properties defined in Section 4.2: {'latitude', 'longitude', 'day', 'hour', 'minute', 'duration', 'remaining time', 'battery consumption (%)', 'on-peak hour'}. This is followed by plotting the t-SNE results on a 2-D plane as seen in Fig. 9.

It can be clearly seen that in each cluster, it would be challenging and complicated to distinguish legitimate tasks from illegitimate tasks, which fulfills the requirement of the adversarial modelling in this study.

4.3.5. Discussions

As shown in Section 4.3.2, only slight increases can be observed in most performance metrics when only the constrained approach is applied. A possible reason can be told from the city map (shown in Fig. 8(a)) where many user events gather in one area, downtown. A small city has a relatively small downtown, which implies that some neurons in a trained SOFM in the city will be close to each other and will be discarded during the selection, as shown in Fig. 8(c). However, these neurons

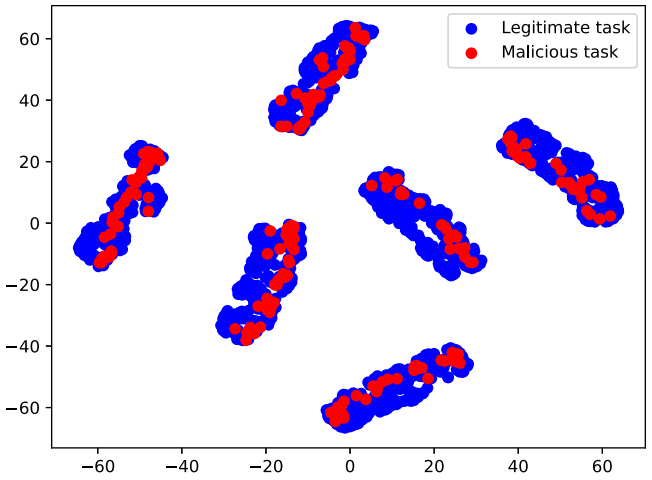


Fig. 9. t-SNE analysis to demonstrate the challenge of distinguishing legitimate tasks from illegitimate tasks.

located in downtown can cover significantly more user events than those not in the urban area. The benefits of enforcing non-overlapping coverage can then be canceled by the difference between the neuron quality.

5. Conclusions

Fake task submissions to Mobile Crowdsensing (MCS) platforms introduce serious threats to the battery lifetime of the participant devices, as well as resource utilization of the MCS servers. The lack of data regarding malicious task submissions makes the development of defense mechanisms rather challenging. To this end, we have proposed new models for fake task submission attacks by improving the impact of attack center determination and examined them with experiments in two cities. The proposed attack models leverage Self-Organizing Feature Maps (SOFM) to build a locally reconfigurable SOFM to determine the centers and other features of fake/illegitimate tasks submitted to MCS platforms. Based on the numerical results, we have concluded that the proposed refined scheme for locally reconfigurable SOFM significantly outperforms the previously proposed SOFM-based fake task models. More specifically, when compared to the previously proposed SOFM-based fake task model, the proposed locally reconfigured SOFM architecture can achieve up to 139% improvements in terms of the impacted participants in an MCS system.

The proposed model is currently being extended to incorporate real time user events rather than uniform distribution in the simulator.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Liu, H. Shen, H.S. Narman, W. Chung, Z. Lin, A survey of mobile crowdsensing techniques: a critical component for the internet of things, ACM Trans. Cyber-Phys. Syst. 2 (3) (2018).
[2] R.K. Ganti, F. Ye, H. Lei, Mobile crowdsensing: current state and future challenges, IEEE Commun. Mag. 49 (11) (2011) 32–39, doi:10.1109/MCOM.2011.6069707.
[3] F. Restuccia, N. Ghosh, S. Bhattacharjee, S.K. Das, T. Melodia, Quality of information in mobile crowdsensing: survey and research challenges, ACM Trans. Sen. Netw. 13 (4) (2017).
[4] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, P. Bouvry, A survey on mobile crowdsensing systems: challenges, solutions, and opportunities, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2419–2465.
[5] M. Pouryazdan, B. Kantarci, T. Soyata, H. Song, Anchor-assisted and vote-based trustworthiness assurance in smart city crowdsensing, IEEE Access 4 (2016) 529–541, doi:10.1109/ACCESS.2016.2519820.
[6] C. Fiandrino, F. Anjomshoa, B. Kantarci, D. Kliazovich, P. Bouvry, J.N. Matthews, Sociability-driven framework for data acquisition in mobile crowdsensing over fog computing platforms for smart cities, IEEE Trans. Sustain. Comput. 2 (4) (2017) 345–358.
[7] O. Alvear, C.T. Calafate, J.-C. Cano, P. Manzoni, Crowdsensing in smart cities: overview, platforms, and environment sensing issues, Sensors 18 (2) (2018) 460.
[8] Z. Peng, X. Gui, J. An, T. Wu, R. Gui, Multi-task oriented data diffusion and transmission paradigm in crowdsensing based on city public traffic, Comput. Netw. 156 (2019) 41–51.
[9] F. Khan, A.U. Rehman, J. Zheng, M.A. Jan, M. Alam, Mobile crowdsensing: a survey on privacy-preservation, task management, assignment models, and incentives mechanisms, Fut. Gener. Comput. Syst. 100 (2019) 456–472.
[10] L. Xiao, D. Jiang, D. Xu, W. Su, N. An, D. Wang, Secure mobile crowdsensing based on deep learning, China Commun. 15 (10) (2018) 1–11, doi:10.1109/CC.2018.8485464.

[11] M. Li, Y. Sun, H. Lu, S. Maharjan, Z. Tian, Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems, IEEE Internet Things J. (2019) 1, doi:10.1109/JIOT.2019.2962914.
[12] N. Ruan, L. Gao, H. Zhu, W. Jia, X. Li, Q. Hu, Toward optimal DoS-resistant authentication in crowdsensing networks via evolutionary game, in: Proceedings of the 2016 IEEE Thirty-sixth International Conference on Distributed Computing Systems (ICDCS), 2016, pp. 364–373, doi:10.1109/ICDCS.2016.66.
[13] D. Yang, G. Xue, X. Fang, J. Tang, Incentive mechanisms for crowdsensing: crowdsourcing with smartphones, IEEE/ACM Trans. Netw. 24 (3) (2016) 1732–1744.
[14] L. Xiao, Y. Li, G. Han, H. Dai, H.V. Poor, A secure mobile crowdsensing game with deep reinforcement learning, IEEE Trans. Inf. Forensics Secur. 13 (1) (2018) 35–47, doi:10.1109/TIFS.2017.2737968.
[15] X. Chen, M. Liu, Y. Zhou, Z. Li, S. Chen, X. He, A truthful incentive mechanism for online recruitment in mobile crowd sensing system, Sensors 17 (1) (2017) 79.
[16] S. Gisdakis, T. Giannetsos, P. Papadimitratos, Security, privacy, and incentive provision for mobile crowd sensing systems, IEEE Internet Things J. 3 (5) (2016) 839–853.
[17] B. Hoh, T. Yan, D. Ganesan, K. Tracton, T. Iwuchukwu, J.-S. Lee, Trucentive: a game-theoretic incentive platform for trustworthy mobile crowdsourcing parking services, in: Proceedings of the 2012 Fifteenth International IEEE Conference on Intelligent Transportation Systems, IEEE, 2012, pp. 160–166.
[18] M. Pouryazdan, B. Kantarci, TA-CROCS: trustworthiness-aware coalitional recruitment of crowd-sensors, in: Proceedings of the IEEE Global Communications Conference (GLOBECOM), IEEE, 2018.
[19] M. Tomasoni, A. Capponi, C. Fiandrino, D. Kliazovich, F. Granelli, P. Bouvry, Why energy matters? profiling energy consumption of mobile crowdsensing data collection frameworks, Pervasive Mob. Comput. 51 (2018) 193–208.
[20] Y. Zhang, B. Kantarci, Invited paper: Ai-based security design of mobile crowdsensing systems: review, challenges and case studies, in: Proceedings of the Thirteenth IEEE International Conference on Service-Oriented System Engineering, SOSE 2019, Tenth International Workshop on Joint Cloud Computing, JCC 2019 and 2019 IEEE International Workshop on Cloud Computing in Robotic Systems, CCRS 2019, IEEE, 2019, pp. 17–26.
[21] Y. Zhang, M. Simsek, B. Kantarci, Self organizing feature map for fake task attack modelling in mobile crowdsensing, in: Proceedings of the IEEE Global Communications Conference (GLOBECOM), 2019. Waikoloa, Hawaii, USA.
[22] Y. Zhang, M. Simsek, B. Kantarci, Machine learning-based prevention of battery-oriented illegitimate task injection in mobile crowdsensing, in: Proceedings of the ACM Workshop on Wireless Security and Machine Learning, ACM, Miami, FL, USA, 2019, pp. 31–36.
[23] Y. Zhang, M. Simsek, B. Kantarci, Ensemble learning against Adversarial AI-driven fake task submission in Mobile Crowdsensing, in: Proceedings of the IEEE International Conference on Communications (ICC), 2020. Dublin, Ireland.
[24] T. Kohonen, Self-organized formation of topologically correct feature maps, Biol. Cybern. 43 (1) (1982) 59–69.
[25] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[26] J. Kangas, T. Kohonen, J. Laaksonen, Variants of self-organizing maps, IEEE Trans. Neural Netw. 1 (1) (1990) 93–99.
[27] T. Kohonen, Essentials of the self-organizing map, Neural Netw. 37 (2013) 52–65, doi:10.1016/j.neunet.2012.09.018.
[28] T. Kohonen, E. Oja, O. Simula, A. Visa, J. Kangas, Engineering applications of the self-organizing map, Proc. IEEE 84 (10) (1996) 1358–1384.
[29] C.F.F. Karney, Algorithms for geodesics, J. Geod. 87 (1) (2013) 43–55, doi:10.1007/s00190-012-0578-z.
[30] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, S. Giordano, Crowdsensim: a simulation platform for mobile crowdsensing in realistic urban environments, IEEE Access 5 (2017) 3490–3503.
[31] F. Montori, E. Cortesi, L. Bedogni, A. Capponi, C. Fiandrino, L. Bononi, CrowdSenSim 2.0: a stateful simulation platform for mobile crowdsensing in smart cities, in: Proceedings of the Twenty-second International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, in: MSWIM '19, ACM, Miami Beach, FL, USA, 2019, pp. 289–296.
[32] Y. Zhang, M. Simsek, B. Kantarci, Empowering self-organized feature maps for ai-enabled modelling of fake task submissions to mobile crowdsensing platforms, IEEE Internet Things J. (2020) 1.
[33] M. Simsek, B. Kantarci, Artificial intelligence-empowered mobilization of assessments in COVID-19-like pandemics: a case study for early flattening of the curve, Int. J. Environ. Res. Public Health 17 (10) (2020) 3437, doi:10.3390/ijerph17103437.
[34] Y. Zhang, M. Simsek, B. Kantarci, Empowering self-organized feature maps for AI-enabled modelling of fake task submissions to mobile crowdsensing platforms, IEEE Internet Things J. (2020) 1, doi:10.1109/JIOT.2020.3011461.
[35] Managing costs with time-of-use rates | Ontario Energy Board.
[36] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.